Technical Cybersecurity

The victim

Don’t Do This!

Significant Flaw

- Not checking input!

Makefile options

- -z execstack
- -f no-stack-protect
- -m32
- -no-pie

```c
#include <string.h>

#define BUF_SIZE 5

void smash(char* arg) {
  char buffer[BUF_SIZE];
  strcpy(buffer, arg);
}

int main(int argc, char* argv[]) {
  char* arg = argv[1];
  smash(arg);
  return 0;
}
```

```
CC=gcc
OBJ=function2.o function-args.o print.o err.o err2.o smash.o g
CC_FLAGS=-no-pie -m32 -g -z execstack -fno-stack-protector

%.o: %.c
	$(CC) -c -o $@ $< $(CC_FLAGS)

main: $(OBJ)
	$(CC) -o getenv getenv.o $(CC_FLAGS)
	$(CC) -o smash smash.o $(CC_FLAGS) $(ALT_CC_FLAGS)
	$(CC) -o f2 function2.o $(CC_FLAGS)
	$(CC) -o fa function-args.o $(CC_FLAGS)
	$(CC) -o print print.o $(CC_FLAGS)
	$(CC) -o err err.o $(CC_FLAGS)
	$(CC) -o err2 err2.o $(CC_FLAGS)

clean:
	rm *.o f2 print err err2 fa smash getenv
	rm -rf a.out core
```

Compile & Look

Add 32-bit support

- sudo apt install gcc-multilib

Make smash!

- (remove the **getenv** program reference for now)
- -m32

```
cclamb@ubuntu:~/Work/abi-playground $ gdb smash
Reading symbols from smash...done.
(gdb) disas main
Dump of assembler code for function main:
   0x08048451 <+0>:     lea    ecx,[esp+0x4]
   0x08048455 <+4>:     and    esp,0xfffffff0
   0x08048458 <+7>:     push   DWORD PTR [ecx-0x4]
   0x0804845b <+10>:    push   ebp
   0x0804845c <+11>:    mov    ebp,esp
   0x0804845e <+13>:    push   ecx
   0x0804845f <+14>:    sub    esp,0x14
   0x08048462 <+17>:    call   0x8048492 <__x86.get_pc_thunk.ax>
   0x08048467 <+22>:    add    eax,0x1b99
   0x0804846c <+27>:    mov    eax,ecx
   0x0804846e <+29>:    mov    eax,DWORD PTR [eax+0x4]
   0x08048471 <+32>:    mov    eax,DWORD PTR [eax+0x4]
   0x08048474 <+35>:    mov    DWORD PTR [ebp-0xc],eax
   0x08048477 <+38>:    sub    esp,0xc
   0x0804847a <+41>:    push   DWORD PTR [ebp-0xc]
   0x0804847d <+44>:    call   0x8048426 <smash>
   0x08048482 <+49>:    add    esp,0x10
   0x08048485 <+52>:    mov    eax,0x0
   0x0804848a <+57>:    mov    ecx,DWORD PTR [ebp-0x4]
   0x0804848d <+60>:    leave
   0x0804848e <+61>:    lea    esp,[ecx-0x4]
   0x08048491 <+64>:    ret
End of assembler dump.
(gdb) disas smash
Dump of assembler code for function smash:
   0x08048426 <+0>:     push   ebp
   0x08048427 <+1>:     mov    ebp,esp
   0x08048429 <+3>:     push   ebx
   0x0804842a <+4>:     sub    esp,0x14
   0x0804842d <+7>:     call   0x8048492 <__x86.get_pc_thunk.ax>
   0x08048432 <+12>:    add    eax,0x1bce
   0x08048437 <+17>:    sub    esp,0x8
   0x0804843a <+20>:    push   DWORD PTR [ebp+0x8]
   0x0804843d <+23>:    lea    edx,[ebp-0xd]
   0x08048440 <+26>:    push   edx
   0x08048441 <+27>:    mov    ebx,eax
   0x08048443 <+29>:    call   0x80482e0 <strcpy@plt>
   0x08048448 <+34>:    add    esp,0x10
   0x0804844b <+37>:    nop
   0x0804844c <+38>:    mov    ebx,DWORD PTR [ebp-0x4]
   0x0804844f <+41>:    leave
   0x08048450 <+42>:    ret
End of assembler dump.
(gdb)
```

```
(gdb) r
Starting program: /home/cclamb/Work/abi-playground/smash AA
[Inferior 1 (process 89655) exited normally]
(gdb) r AAAAA
Starting program: /home/cclamb/Work/abi-playground/smash AAAAA
[Inferior 1 (process 89713) exited normally]
(gdb) r AAAAAA
Starting program: /home/cclamb/Work/abi-playground/smash AAAAAA
[Inferior 1 (process 89751) exited normally]
(gdb) r AAAAAAAAAA
Starting program: /home/cclamb/Work/abi-playground/smash AAAAAAAAAA
[Inferior 1 (process 89795) exited normally]
(gdb) r AAAAAAAAAAA
Starting program: /home/cclamb/Work/abi-playground/smash AAAAAAAAAAA
[Inferior 1 (process 89825) exited normally]
(gdb) r AAAAAAAAAAAA
Starting program: /home/cclamb/Work/abi-playground/smash AAAAAAAAAAAA
[Inferior 1 (process 89840) exited normally]
(gdb) r AAAAAAAAAAAAA
Starting program: /home/cclamb/Work/abi-playground/smash AAAAAAAAAAAAA
```

Find the overflow

Pretty easy…

Not always this easy.

The buffer is five bytes

- Overflow doesn't cause problems until we've entered 13
- Why?

Overflow starts at **FIVE**

- String has terminating NULL -> SIX bytes!
- We don't hit anything that matters until the 13th byte

```
(gdb) disas smash
Dump of assembler code for function smash:
   0x08048426 <+0>:     push   ebp
   0x08048427 <+1>:     mov    ebp,esp
   0x08048429 <+3>:     push   ebx
   0x0804842a <+4>:     sub    esp,0x14
   0x0804842d <+7>:     call   0x8048492 <__x86.get_pc_thunk.ax>
   0x08048432 <+12>:    add    eax,0x1bce
   0x08048437 <+17>:    sub    esp,0x8
   0x0804843a <+20>:    push   DWORD PTR [ebp+0x8]
   0x0804843d <+23>:    lea    edx,[ebp-0xd]
   0x08048440 <+26>:    push   edx
   0x08048441 <+27>:    mov    ebx,eax
   0x08048443 <+29>:    call   0x80482e0 <strcpy@plt>
   0x08048448 <+34>:    add    esp,0x10
   0x0804844b <+37>:    nop
   0x0804844c <+38>:    mov    ebx,DWORD PTR [ebp-0x4]
   0x0804844f <+41>:    leave
   0x08048450 <+42>:    ret
End of assembler dump.
(gdb) b *0x0804844b
Breakpoint 2 at 0x804844b: file smash.c, line 8.
(gdb) r AAAAAAAAAAAAA
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/cclamb/Work/abi-playground/smash AAAAAAAAAAAAA

Breakpoint 2, smash (arg=0xffffd1de 'A' <repeats 13 times>) at smash.c:8
8       }
(gdb) x/20xw $esp
0xffffcee0:     0x00000009      0xffffd1b7      0x41e0f849      0x41414141
0xffffcef0:     0x41414141      0x41414141      0xffffcf00      0x08048482
0xffffcf00:     0xffffd1de      0x00000000      0xffffcfe0      0x08048467
0xffffcf10:     0x00000002      0xffffcfd4      0xffffcfe0      0xffffd1de
0xffffcf20:     0xf7fe59b0      0xffffcf40      0x00000000      0xf7df7e81
(gdb)
```

```
(gdb) r AAAAAAAAAAAAABBBBCCCC
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/cclamb/Work/abi-playground/smash AAAAAAAAAAAAABBBBCCCC

Breakpoint 2, smash (arg=0xffffd100 "\003") at smash.c:8
8       }
(gdb) x/20xw $esp
0xffffced0:     0x00000009      0xffffd1af      0x41e0f049      0x41414141
0xffffcee0:     0x41414141      0x41414141      0x42424242      0x43434343
0xffffcef0:     0xffffd100      0x00000000      0xffffcfd0      0x08048467
0xffffcf00:     0x00000002      0xffffcfc4      0xffffcfd0      0xffffd1d6
0xffffcf10:     0xf7fe59b0      0xffffcf30      0x00000000      0xf7df7e81
(gdb)
```

MOAR OVERWRITE

Now we have some idea where we want to put things.

Let’s fill in the blanks.